第1章 64点高速フーリエ変換回路設計のポイント

—— Design Wave設計コンテスト2007 総評

和田知久，田中孝一，林 輝彦

本誌2007年5月号において，Design Wave設計コンテスト2007の入賞者を発表した．ここでは，課題に対する回路設計上のポイントを解説する．Professional部門の入賞者を決定するにあたっては，技術的な評価のために，エンジニアの方々にご協力いただいた．その審査委員である田中孝一氏と林輝彦氏に，Professional部門の総評をお願いした．もう一方のStudent部門については，琉球大学にご協力いただき，同大学共催の「LSIデザインコンテスト2007」を同部門とさせていただいている．主催者であり，かつコンテスト課題作成者でもある和田知久氏に，Student部門の総評をお願いした．

（編集部）

1．コンテストの課題と設計のポイント

和田知久

設計テーマは，ディジタル信号処理では必ず登場する高速フーリエ変換（FFT：fast Fourier transform）回路の設計です．高速フーリエ変換は，離散フーリエ変換（DFT：discrete Fourier transform）を高速に計算する手法です．

計算式自体は，式（1）に示すように単純です．

$$X(k)=\sum_{n=0}^{N-1}x(n)\cdot e^{-j\left(\frac{2\pi}{N}\right)nk}\quad(k=0,\ 1,\ \cdots,\ N-1)\quad\cdots(1)$$

今回はIEEE 802.11a/gなどのワイヤレスLANでよく使われている64点のFFT，すなわち$N=64$です．

実は高速フーリエ変換回路は非常にポピュラな課題であり，参加者の皆さんが同じような設計をしてくるのではないかと心配していました．しかし，予想に反してそれぞれの技量や経験を生かしたオリジナリティあふれる設計が投稿され，コンテストとして大いにもりあがりました．

実際に64点FFTを計算する回路を構成する場合，効率が良く，ポピュラな方法のひとつは，$64=4^3$という性質を利用して，3回の計算ステージに分けて計算を実行するものです．

式（2）に示すように，インデックスnをmod 4の数n_i，k_i（$i=0$，…，3）を用いて変数変換すると，3重のΣを用いた式に変形できます．この1重のΣの計算は単純化することができ，いわゆるRADIX-4演算ステージになります．

$$X(k)=\sum_{n=0}^{63}x(n)\cdot W_{64}^{nk}$$
$$k=k_0+4k_1+16k_2\quad(k_0,\ k_1,\ k_2=0\sim3)$$
$$n=n_0+4n_1+16n_2\quad(n_0,\ n_1,\ n_2=0\sim3)$$
$$X(k)=\sum_{n_0=0}^{3}\sum_{n_1=0}^{3}\sum_{n_2=0}^{3}x(n_0+4n_1+16n_2)W_{64}^{(n_0+4n_1+16n_2)(k_0+4k_1+16k_2)}\quad\cdots(2)$$

従って，図1に示すように，RADIX-4演算ステージを3段用いるアーキテクチャによりFFT計算を実行できます．ただし，変数変換の都合により，第3ステージの出力の値は本来必要とする順序で計算が出力されないので，式（3）に示す順番の入れ替え（リオーダ）が必要となります．

$$X(k_0+4k_1+16k_2)=x3(k_2+4k_1+16k_0)\quad\cdots(3)$$

KeyWord 高速フーリエ変換，FFT，ワイヤレスLAN，RADIX-4，バタフライ演算，マイクロプログラム方式，DSP，リオーダ処理，FPGA
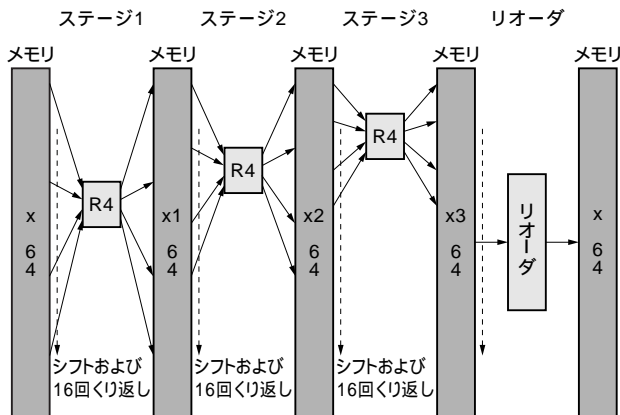
図1 RADIX-4の64点FFT演算アーキテクチャ
3段の演算の後，リオーダを行う．R4はRADIX-4のバタフライ演算．

すなわち，3段のRADIX-4ステージと1段のリオーダ・ステージで64点高速フーリエ変換を実現することができます．設計のポイントとしては，

- RADIX-4アーキテクチャを採るのか，別のアルゴリズムで計算するのか
- 各ステージで必要なメモリをどのように実現するのか
- 各ステージで似た回路が多いので，どのように回路を最適化するのか

などがあげられました．

2. Professional部門 総評

田中孝一，林 輝彦

Design Wave 設計コンテスト 2007では，本誌において完成度の高い詳細なリファレンス・デザインが示されていたため，おおむねこのアイデアに従う設計が応募され，設計者ごとに個性を打ち出すのは難しいのではないかと予想していましたが，意に反して個性的な設計が応募され，枯れた技術テーマであると思われているFFT演算が，実は奥の深いテーマであるという印象を受けました．

● 審査結果

「Design Wave 設計コンテスト 2007」Professional部門の入賞者は，次の方に決定しました．

- 1位 石井康雄
- 2位 匿名
- 3位 チームきろんぼ（稲垣博彦，音田良博，篠原慈明）

（敬称略）

賞品として，第1位の石井氏には，発表会講演を兼ねた2泊3日の沖縄旅行のほか，副賞のWindows Vista対応ノート・パソコン（NEC LaVie L PC-LL750HG）が，第2位には32V型液晶テレビ（三菱 LCD-H32MX60）が，第3位のチームきろんぼには，デジタル・カメラ（ニコン COOLPIX L11）が贈られました．

● 審査のポイント

信号処理アルゴリズムと設計手法のいずれか，または両方に普遍性があることを最大の評価尺度としています．

優れたアルゴリズムや設計手法というのは，真空管やトランジスタ，さらに将来出現するであろう素子など，技術を実現する具体的なデバイスが変わろうと，それに依存せずに各時代において優位性が保たれると考えられるのです．つまり，後世の人類にとっても優れていると認められるもの，すなわち，歴史的な価値があると思われるものに高い評価を与えました．もちろん，普遍的な価値のあるアイデアによって設計されたLSIは，結果として，2007年の時点においても，信号処理速度や回路規模など（おそらく消費電力も）の定量的な性能パラメータを最適化するものとなっています．特定のメーカの特定の半導体，特定の開発言語などに依存せず，普遍的に優れているものを創出しようという高い志は，現時点で実現し得る最適化された仕様となって現れています．

既存のものとは異なる新しいアイデアを抱いたとき，それが既存のものを代替するような優れた素性を持つ独創的なアイデアなのか，それとも，技術的な限界があり，いかなるチューニングを試みても既存のものを凌ぐことはできず，歴史的には葬りさられてしまうような単なる奇抜なアイデアなのかを，客観的に自己評価して設計をした方が，高い評価を獲得しました．

今回のコンテストの評価尺度とは直接的には関係ありませんが，信号処理速度や回路規模などの性能パラメータが同一であるならば，信号処理アルゴリズムや回路の対称性などにも着目して，美しい設計を心がけていただきたいと思います．

今回のコンテストのテーマである64点FFT回路では，仕様書（本誌2006年11月号，pp.143-155を参照）で示された回路は，基数4のバタフライ演算を採用し，処理を3個のステージに分割しています．

図2に仕様書で示された回路のステージ1を示します．
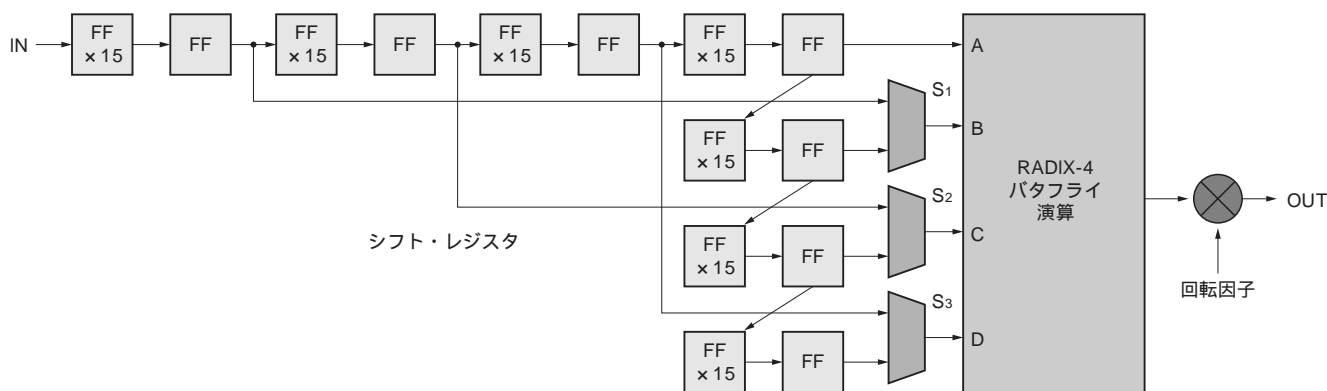
図2　ステージ1の処理
仕様書で示された回路である．ステージ2，ステージ3も，シフト・レジスタの長さ以外は，まったく同じ構成を取り，同じバタフライ演算を用いている．

ステージ2，ステージ3も，シフト・レジスタの長さ以外は，まったく同じ構成を取り，同じバタフライ演算を用いています．

このように，この回路は美しい対称性があるため，信号の流れにボトル・ネックとなる部分がなく，比較的簡単な回路で高速な動作を実現できることが容易に想像できます．また，美しい対称性を有する回路をソフトウェアで実現した場合には，同一の関数を複数の回路で共有できるため，より少ないコード量で信号処理を実行できるでしょう．

優れた性能を有する設計は，自然現象の本質を見抜き，それを最大限に利用しているので，必然的に自然界の持つ対称性などの美しさを反映したものになるということを常に意識して設計技術を磨いてください．

● 入賞した設計はオリジナリティがある

今回入賞した3チームは，すべて，仕様書で示された回路に対して新規性を打ち出そうと努力した跡がうかがえます．

第1位の石井氏と第3位のチームきろんぼは，LSI内部における信号処理をソフトウェアで実現する，いわゆる，マイクロプログラム方式（厳密に定義され一般的に用いられている用語ではなく，応募者もこの用語を用いていない）を採用しました．また，第2位の設計は，独自のFFT信号処理アルゴリズムを考案し，RADIX-4方式に対する優位性を達成しようと試みています．

● 演算量が少ないと考えられるアルゴリズムを採用

第1位の設計は，既存のFFT演算アルゴリズムを十分に調査し，現時点で最も演算量が少ないと考えられるアルゴリズムを採用しました．さらに，一般的なディジタル信号処理ではなく，このFFT演算に特化したアーキテクチャ（乗算器と加算器の個数の比を1:5とする）を有するプロセッサ（＝データ・パス）を構築することによって，ハードウェア資源の利用効率の最適化を図っています．また，演算の制御は自動生成されるステート・マシンで行っています．

第1位の石井氏は，信号処理アルゴリズムにおいて新たな提案は行なっていませんが，ハードウェア資源の利用効率を最適化しただけではなく，演算を制御するステート・マシンを独自のソフトウェアによって自動生成することにより，設計を容易に拡張することを可能として設計のIPコア化を図っており，徹底的に普遍性を追求しようという高い志が伝わってきました．石井氏の設計は，今回の入賞者の中で一歩抜きん出たものでした．

● DSPによりFFT演算を行う

第3位の設計は，マイクロプログラム方式を採用している点においては，第1位の設計と同様です．しかし，一般的なハーバード・アーキテクチャを有するDSP（digital signal processor）を構築してFFT演算を実行しているため，ハードウェア資源の利用効率が最適化されておらず，第1位の設計と性能面で大きな差が付きました．実際に両者を比較すると，第1位の設計は第3位の設計と比べて，回路規模が2倍程度であるにも関わらず，処理できる信号の標本化周波数が100倍程度に上がっています．

● バタフライ演算を用いないアルゴリズム

第2位の設計では，バタフライ演算を用いず帰還ループとシフト・レジスタを利用する独自のFFT信号処理アル
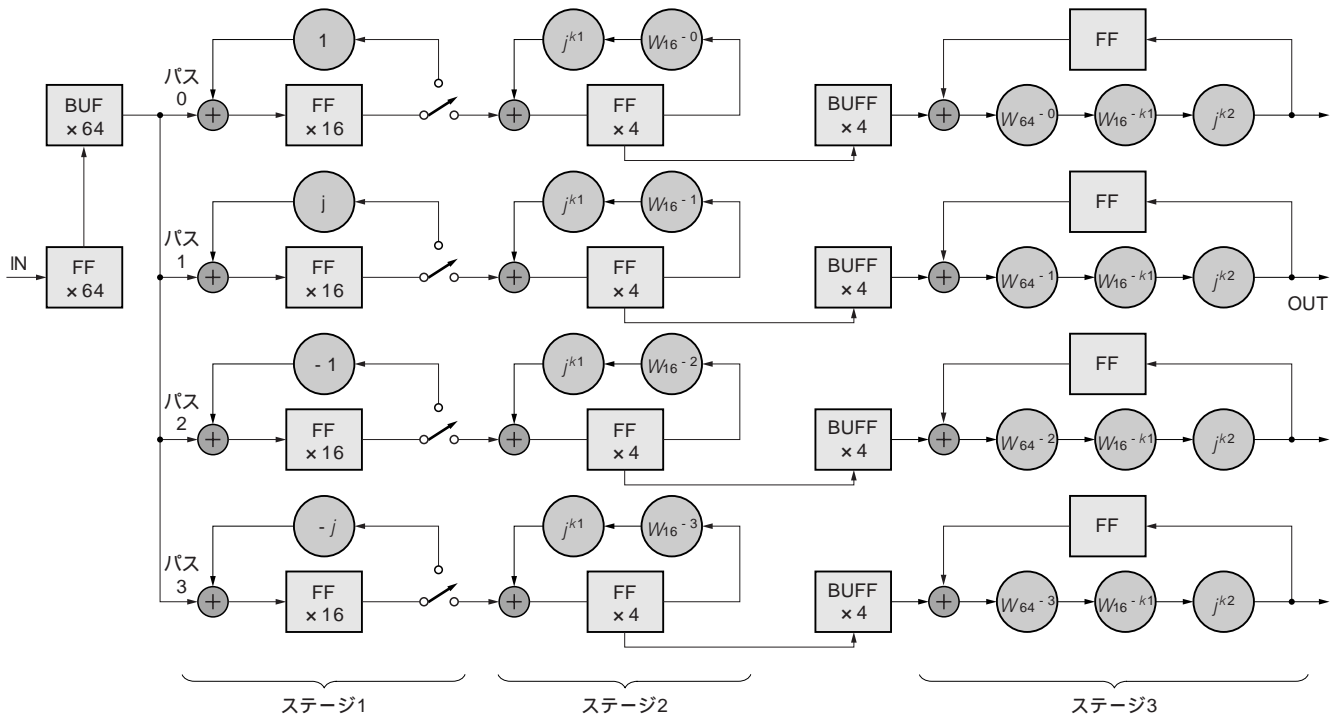
図3　バタフライ演算を用いないFFT演算のアーキテクチャ
第2位の設計である．帰還ループとシフト・レジスタを利用する．回転因子ROMと複素数変数同士の乗算器を不要にすることで，回路規模の縮小を目指した．

ゴリズムを考案し実装しています．従来の方法と演算量は同程度であるものの，回転因子ROMと複素数変数どうしの乗算器を不要として，回路規模を縮小する可能性を示しています．

このアルゴリズムにもとづく回路を**図3**に，タイミング・チャートを**図4**に示します．

時刻0～255において，入力データ$x(0)$～$x(63)$は，基本クロックに同期して，FF×64に入力されます．これらは，BUF×64にコピーされ，そこに256内部クロック(時刻256～511)保持されます．その間，これら64個のデータが4個のパスに並列に，4回反復して送出されます．

ステージ1では，16段のシフト・レジスタFF×16と帰還ループを用いて，64内部クロックにデータに定数を乗算しながら累積します(ただし，最初の16内部クロックでは帰還ループにはゼロが入力され，FF×16の出力はステージ2に送られ，次の48内部クロックでのみ帰還ループを用いる)．この64内部クロック(時刻256～319)におけるFF×16の累積結果がステージ1の出力$x1(0+16k_0)$～$x1(15+16k_0)$[注1]($k_0=0, 1, 2, 3$；パス番号)，すなわちステージ2の入力となります(**図4**においては，$x1(0)$～$x1(15)$と略記)．

ステージ2では，4段のシフト・レジスタFF×4と帰還ループを用いて，16内部クロックにデータを乗算しながら累積します(ただし，処理の開始時点で，FF×4は，クリアされる)．この16内部クロック(時刻320～335)におけるFF×4の累積結果がステージ2の出力$x_2(0+4k_1+16k_0)$～$x2(3+4k_1+16k_0)$[注1]，すなわち，ステージ3の入力となります(**図4**においては，$x_2(0)$～$x_2(3)$と略記)．

この入力データはステージ2とステージ3の間にある4ワードのBUFF×4に転写され，ここに64内部クロック保持されます．次の48内部クロックは，ステージ1からの入力データがないので，ステージ2は休んでいます(**図4**のステージ2入力，時刻336～383)．つまり，ステージ2の稼働率は，1/4です．ステージ2では，パラメータk_1の値を$k_1=0, 1, 2, 3$と切り替え，この64内部クロックの処理を反復します．なお，BUFF×4が書き換えられるタイミング(64内部クロックごと)に合わせて，ステージ3で用いるパラメータk_1の値を$k_1=0, 1, 2, 3$と切り替えます．

ステージ1とステージ2の処理から分かるように，内部クロック周波数の1/4が基本クロック周波数(信号処理レー

注1：この$x1$，$x2$は，本誌2006年11月号における$x1$，$x2$と定義が異なる．
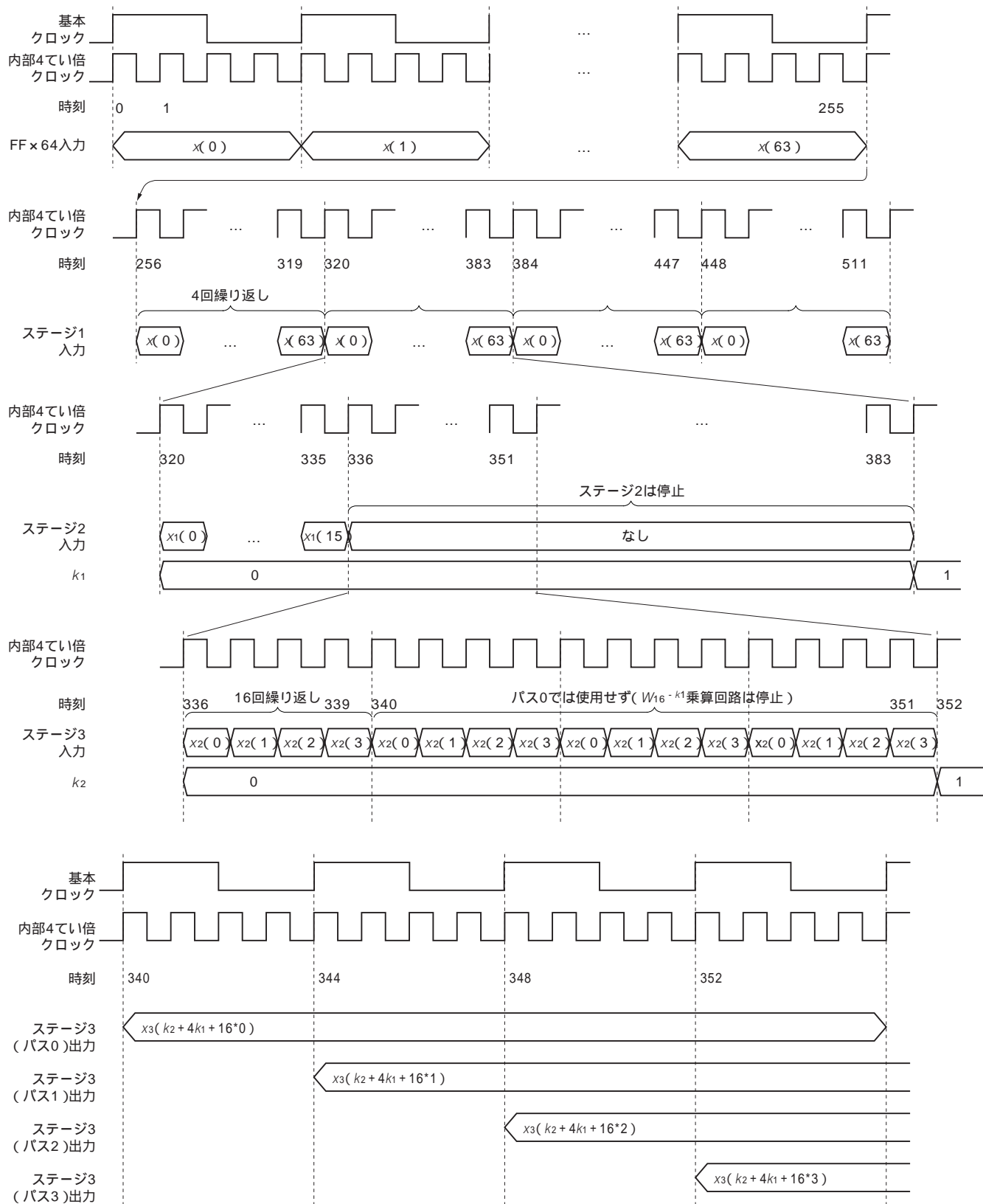
図4　バタフライ演算を用いないFFT演算の動作
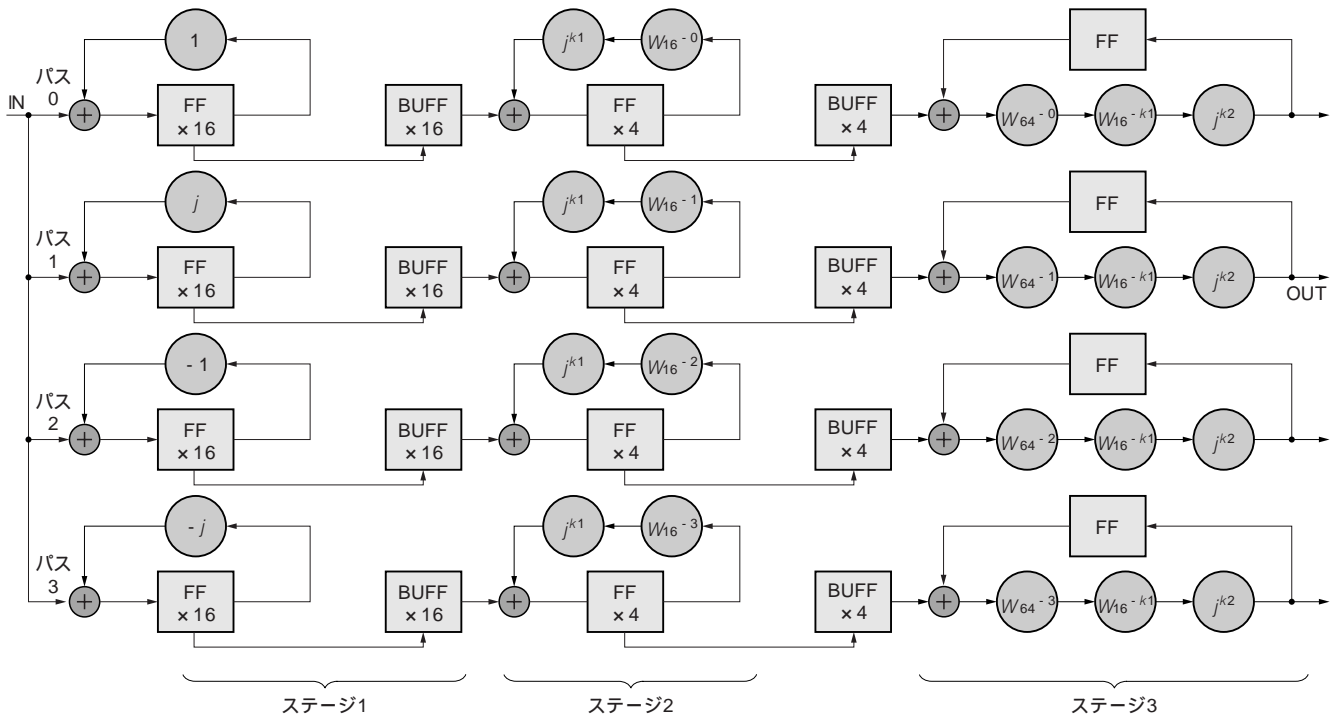
図6 バタフライ演算を用いないFFT演算の改善例
ステージ1の前段にあったFF×64およびBUF×64を廃し，それに代えて，64基本クロックだけデータを保持するBUFF×16をステージ1の後段に設けた．

ト，このFFT演算回路に入力できる信号の標本化周波数）ということになります．

ステージ3では，1ワードの遅延回路FFと帰還ループを用いて，4内部クロックにデータを乗算しながら累積します（ただし，処理の開始時点でFFはクリアされる）．この4内部クロック（時刻336〜339）におけるFFの累積結果がステージ3（パス0）の出力$x3(k_2+4k_1+16*0)$となります．

次の12内部クロックは，ステージ3（パス0）は，休んでいます．ステージ2は1内部クロックあたりに1/4ワードのデータしか出力してこないので，ステージ3の稼働率も必然的に1/4より高くすることはできません．そのため，ステージ3は原理的には内部クロックではなく，基本クロックに同期させて処理を実行すればよいことになります．それに対し，第2位の設計ではステージ3を基本クロックではなく内部クロックに同期させて動作させて（つまり，4倍速で動作させて），ステージ3における定数$W_{16}{}^{-k1}$乗算回路（**図5**）を4個のパスで時分割に利用する（パスiでは，時刻$336+4i$〜$339+4i$で4個のステージ3入力$x2(0)$〜$x2(3)$から$x3(k_2+4k_1+16i)$を計算し，時刻$340+4i$で出力する）ことによって，回路規模の縮小を図っています．16内部クロック（時刻336〜351）で，ステージ3のパス0〜3の計4ワードの出力が出揃うことになりますが，ステージ3では，パラメータk_2の値を，$k_2=0$，1，2，3と切り替えて，この16内部クロックの処理を反復します．
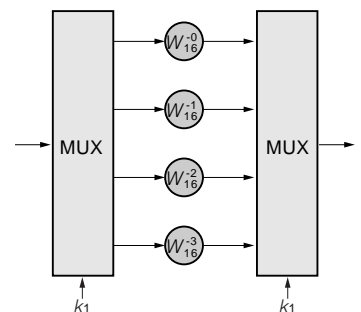
図5
定数$W_{16}{}^{-k1}$乗算回路

第2位の回路のアイデアを基本的に踏襲し，小改良を加えた回路を**図6**に示します．この回路ではステージ1の前段にあったFF×64およびBUF×64を廃し，それに代えて，64基本クロックだけデータを保持するBUFF×16をステージ1の後段に設けました．

リスト1は**図6**の回路の動作をC言語で記述したものです．シミュレーションによって正しく動作することを検証しています．このリストからも分かるように，この方式では後ろのステージほど処理が重くならざるを得ません．

リスト1　バタフライ演算を用いないFFT演算のC言語によるシミュレーション

```
#define N       64
#define PI      3.14159265

typedef         struct   com{
                float re;
                float im;
}COMPLEX;
                int k0,k1,k2;
                COMPLEX w[N]; // table of twiddle factors
                              // w[k] = exp(+j*2*PI*k/N)
                                 (k = 0,1,…,N-1)
                COMPLEX data_i[N]; // input data
                COMPLEX data_o[N]; // output data
                COMPLEX ff16[16],ff4[4],ff;
                                    // shift registers

COMPLEX         cad(COMPLEX x,COMPLEX y);   // x+y
COMPLEX         cml(COMPLEX x,COMPLEX y);   // x*y

void  main(void)
{
      for(k0=0;k0<4;k0++)
      {
                stage1();
                for(k1=0;k1<4;k1++)
                {
                        stage2();
                        for(k2=0;k2<4;k2++)
                        {
                                stage3();
                        }
                }
      }
}


void  stage1(void)
{
      int i,j;
      COMPLEX temp;

      for(j=0;j<16;j++)
      {
                ff16[j].re = 0.0;
                ff16[j].im = 0.0;
      }

      for(i=0;i<4;i++)
      {
                for(j=0;j<16;j++)
                {
                        temp = cml(ff16[j],w[16*k0]);
                        ff16[j] =
                              cad(temp,data_i[16*i+j]);
                }
      }
}

void  stage2(void)
{
      int i,j;
      COMPLEX temp;

      for(j=0;j<4;j++)
      {
                ff4[j].re = 0.0;
                ff4[j].im = 0.0;
      }

      for(i=0;i<4;i++)
      {
                for(j=0;j<4;j++)
                {
                        temp = cml(ff4[j],w[4*k0]);
                        temp = cml(temp,w[16*k1]);
                        ff4[j] = cad(temp,ff16[4*i+j]);
                }
      }
}

void  stage3(void)
{
      int i;
      COMPLEX temp;

      ff.re = 0.0;
      ff.im = 0.0;

      for(i=0;i<4;i++)
      {
                temp = cad(ff,ff4[i]);
                temp = cml(temp,w[k0]);
                temp = cml(temp,w[4*k1]);
                ff = cml(temp,w[16*k2]);
      }

      data_o[16*k2+4*k1+k0] = ff;
}
```

第2位の設計は，アイデアは斬新でしたが，ステージ間の対称性を崩した結果，C言語のソース・リストは読みにくいものになりました．ハードウェアにおいても，ステージ3が明らかなボトル・ネックとなって，回路全体の性能の足を引っ張る結果となっていると考えられます．

提出されたレポートによると，複素数定数と複素数変数の乗算回路（以後，k^*x 回路）は，複素数変数同士の乗算回路（以後，x^*y 回路）に比べて，回路規模がほぼ半分になっているとのことでした．しかし，仕様書で示された回路では，3個の x^*y 回路を用いるのに対して，9個の k^*x 回路が必要になっています．ステージ3で k^*x 回路を省略するように工夫しても，9個の k^*x 回路を用いるのであれば，乗算回路の規模は，必然的に約1.5倍に増えてしまいます．

ほかの入賞者とはまったく異なる斬新な解法の切り口を見せて来ましたが，残念ながら現時点では，回路規模および処理速度などにおいて，従来の技術に対する明確な優位性が達成されているとは考えられませんでした．これはこのアルゴリズムの本質的な問題によるものである可能性が高いと思われます．

● リオーダ処理のアルゴリズム

今回のコンテストの応募者で，リオーダ処理に関して言及している方は，ほとんどいませんでした，その理由は，この処理はかなり単純であるため，ここで独自性を出すことはできなかったからであると言えましょう．単純な規則性に気付かずに，めんどうな処理をしている応募者もいたので，ここに，そのアルゴリズムを示します．

コンテスト仕様書において，リオーダの式［式(1)］とと

もに配列x_3から配列Xを生成するインデックスの対応表が具体的に与えられていました．しかしこの表を見なくとも，リオーダの式だけから，直ちにアルゴリズムを思いついたことでしょう．

式(1)において，配列のインデックスは，実は，3けたの4進数で表現されているのです．これをより明示的にするために，

$$a16+4b+c=(a,\ b,\ c)_4$$

(a，b，cは，0以上3以下の整数)

と表記すれば，式(1)は，

$$X((k_2,\ k_1,\ k_0)_4)=x_3((k_0,\ k_1,\ k_2)_4)$$

と表されます．これより，直ちに，配列x_3のインデックスの4進数表現の左右反転が配列Xのインデックスの4進数表現になっていることが分かります．これらの配列のインデックスを2進数で表現すると6ビット表現になりますが，上述の左右反転は，インデックスの6ビット表現の上位2ビットと下位2ビットの入れ替えにほかならないので，リオーダ処理は容易に実現されます．

3. Student部門 総評

和田知久

2007年3月16日，沖縄県那覇空港近くの沖縄産業支援センター大ホールにて，「第10回LSIデザインコンテストin沖縄2007」発表会・最終審査会を開催しました．本コンテストは学生を対象にハードウェア記述言語用いた半導体集積回路(LSI)の設計を行うコンテストです．決められた共通課題に従いつつ，ある程度の自由度を持ちながらオリジナリティのある集積回路を設計することを特徴とします．課題はなるべく現実的で，かつ，その時節にホットな設計課題を，初心者学生でも対応できるように工夫がされています．

コンテストは今年度が10回目です．昨年までと同様に，コンテスト発表会の前に九州大学 大津留榮佐久氏による「シリコン・組み込みソフトウェア産業」という基調講演と，「沖縄シリコン・組み込みソフトウェア産業の将来像」というテーマの国際フォーラムを開催しました．

ホームページ[(1)]でコンテストの告知を行い，本誌の設計コンテストと協力して，1月26日の締め切りまでに，沖縄県内44名，沖縄県以外の国内23名，海外15名，合計82名の学生の参加がありました．参加団体は琉球大学，東京大学，会津大学，豊橋技術科学大学，千葉大学，京都大学，大分工科短期大学校，九州工業大学，長崎大学より，そして国外よりインドネシアのInstitut Teknologi Bandung，韓国のChosun Universityです．

事前選考の結果，国外の2チームを含む8チームの代表者を沖縄に招待し，琉球大学の1チーム，そして社会人の優勝チーム1チームと合わせて10チームによる発表会を行いました．

● 審査結果

コンテストの仕様書では，RADIX-4ベースで，各ステージ間にシフト・レジスタを用いたアーキテクチャを示していました．しかしFFTはポピュラな演算のため，多数のチームは設計仕様書のアーキテクチャとは異なる方式でオリジナリティを出すように設計してきました．

また，比較的演算量の大きいFFT演算でも十分安価なFPGAにて実装できることもあり，多数のチームがFPGAにて実装し，リアルタイム処理のデモを行いました．FPGAに対して回路を最適に実装する制約と，ASICを前提にする場合で制約条件がまったく異なるので，より一層学生達の設計物を比べることが困難に感じられる発表会でした．

回路設計コンテストらしく，似た機能の回路をまとめることで回路規模を削減したり，三角関数の周期性を利用して演算やメモリを減らすなどのオーソドックスな工夫もありました．

コンテスト発表会の例年の一つ特徴は，ほかのチームの発表に対して弱点をつく質問をしたり，審査員に対してアピールを行うこともあります．

審査の結果，京都大学のチーム日本正彦が優勝を勝ち取り，千葉大学の3連覇を阻止することになりました．しかし発表会には昨年と一昨年で2連覇の若林秀明氏が参加しており，もし若林氏が参加していたらと思いました．Institut Teknologi Bandungからは毎年多数の優秀な応募があります．昨年に続き2年連続で入賞しました．

- 優勝：Outstanding Design Award
  チーム日本正彦(京都大学 修士1年，廣本正之，日向文彦)
- 準優勝：Special Feature Award
  チームまだ見ぬ君の名に『智』の字あれ(千葉大学 修士1

年，根尾 敦，神田康博，滝沢 努）

チームVR46（Institut Teknologi Bandung，4年，Muh Syafiq Irsyadi，Yan Syafri Hidayat，Ade Irawan）

● 学科長奨励賞：Faculty Chair Special Award

琉球大学 修士1年，鄭志安（Zheng Zhi An）

琉球大学2年，本村健太

（敬称略）

● アーキテクチャの工夫

今回はワイヤレスLANなどのポピュラな64点FFTの実装ですが，RADIXの基底をどうするかが一つの選択となりました．

$64 = 4^3$ですので，4入力のRADIX-4演算では3ステージで計算できます．また，RADIX-4方式では4点の複素加減算が必要ですが，その乗算係数は（1，j，-1，$-j$）であり，実際の回路設計では乗算器なしに符号反転や数値の入れ替えで実装できます．このため，一般的にはRADIX-4演算がアーキテクチャ的に優れていると考えられています．

しかしながら，コンテストのふたをあけてみると，最も教科書的なRADIX-2方式やRADIX-8方式を選択したグループが複数ありました．コンテストの趣旨の一つである「人と違ったことをしよう！」に沿った形となりました．

RAXID-8方式では，$64 = 8^2$のように2ステージで計算できるメリットがある反面，乗算・加算が増加したり，平方根の値が必要になるデメリットがあります．各チームはそれぞれ解決方法を提案し，おもしろい発表会となりました．

ちなみに，優勝の京都大学はRADIX-4方式，準優勝の千葉大はRADIX-2方式，Institut Teknologi BandungはRADIX-8方式での入賞でした．

● RAMの削減

FFT演算では，各ステージの演算結果を蓄えるRAMが必要です．

基本課題では64点のデータがシリアルに連続して入力されることを前提にしています．そのために，パイプライン処理を行う場合には，ある64点のシンボル1を処理中に次の64点シンボル2が入力されてきます．従って，各ステージ間のRAMを兼用することはできませんが，あるグループは入力パターンと速度の前提を変えて，上記問題が発生しないことを前提に設計していました．

また，対照的にあるグループは2バンクのRAM構成を前提に，あるシンボル1の計算中にシンボル2が入力され，それと同時に以前に計算が完了しているシンボル0を出力するように，基本課題の仕様に忠実な処理を行っていました．このあたり，どのような前提かを発表会で注意しながら審査が進みました．

● ROMの削減

回転因子（twiddle factor）と呼ばれる，$\cos(x) - j\sin(x)$の値を計算する必要があります．ほとんどのグループは三角関数の4分の1周期のデータを蓄えて，演算させることで，1周期分のデータを作成していました．FFTの演算で用いられる回転因子は64種類の値を出力するように見えるのですが，よくよくFFTアルゴリズムを検討すると，64の値のすべてを計算に使用しないことが分かります．そのようなことまで気がついて回路削減をしていたグループもあり，各チームの努力に頭がさがりました．

● FPGAに特化した設計

今回は入賞できませんでしたが東京大学のチームは，FPGAを前提に，最適化設計を計算レイテンシの最小の設計を発表していました．実は，昨年準優勝チームの後輩にあたるようで，大変良いチャレンジであると感じました．

最近では，FPGAに多量のRAMが内蔵できる時代であり，RAMにテーブルを記憶して，そのRAMの構成方法やアクセス方法に工夫を行ったり，FPGA内にハード・マクロとして搭載されている多量の乗算器を「あるものを使わないのは無駄」という発想で活用した発表もありました．

ここ数年コンテストとしての「ものさし」を大変決めにくくなっており，時代の急激な変化を感じています．

● 今後の予定

このコンテストは来年以降もオープンな形で，また国際的に参加可能な形で，継続していく予定です．優秀チームの代表者を沖縄へ招待し，本年同様に参加者全員にオリジナルTシャツを，優勝・準優勝受賞者に楯を出す予定であり，ハードウェア記述言語を用いたLSIの設計に興味ある学生諸君の参加を期待しています．

琉球大学のある沖縄は本土から感覚的に遠く，このようなチャンスで学生・教官の交流を深めることができ，非常に有意義なイベントと考えています．また，特に今回はイ

表1 発表チーム（敬称略）

	チーム名	メンバ	所 属
1	FM	吉川優也 三留健二	会津大学 学部3年
2	MSIC（韓国）	CheolWon Kang Chang hui BAEK Seok Ho SEO	Chosun University （韓国） 学部生
3	ノンアルコール	川内保奈 三田倫史	大分県立工科短期大学校 学部2年
4	VR46 （インドネシア）	Muh Syafiq Irsyadi Yan Syafri Hidayat Ade Irawan	Institut Teknologi Bandung （インドネシア） 学部生
5	鄭志安（中国）	鄭志安	琉球大学 理工学研究科 修士1年
6	kitune-HW	伊藤康宏 林崎弘成	東京大学大学院 情報理工学系研究科 修士1年
7	石原軍団	杉本憲明 濱田 悠 水上貴史	千葉大学大学院 自然科学研究科 修士1年
8	日本正彦	廣本正之 日向文彦	京都大学大学院 通信情報システム専攻 修士1年
9	まだ見ぬ君の名に 「智」の字あれ	根尾 敦 神田康博 滝沢 努	千葉大学大学院 自然科学研究科 修士1年
10	社会人代表	石井康雄	NEC

表2 審査員（敬称略）

	所 属	名 前
審査委員長	東京大学	藤田昌宏教授
委員	ソニー	木村 睦
委員	大阪大学	谷口研二教授
委員	九州工業大学	黒崎正行助教
委員	Institut Teknologi Bandung	Sarwono Sutikno教授
委員	Chosun University	Kang-Hyeon Rhee教授
委員	沖縄工業高等専門学校	鈴木龍司教授
委員	MGウェーブ	鈴木健一
委員	ソリトンシステムズ	林 輝彦
委員	CQ出版	西野 直樹
委員	マグナデザインネット	和宇慶 康

ンドネシアと韓国からの多数の投稿があり，インドネシアと韓国の学生・教官の参加をいただきました．沖縄が地理的にほかのアジアの国々に近く，今後アジアとの連携を行っていくことが非常に重要であるということを体験を通して学べたように思います．沖縄に遊びに来たいという不純な動機も含めて歓迎しますので，ドシドシ参加をお願いします．

なお，このコンテストは，主催の琉球大学LSIデザインコンテスト実行委員会，共催の琉球大学工学部情報工学科，沖縄産業支援センター，フロム沖縄推進機構，九州半導体イノベーション協議会，協賛のソニーLSIデザインにより実施されています．

最後に，**表1**にコンテスト発表会の参加者を，**表2**に審査員の一覧をまとめておきます．

参考・引用*文献

(1) http://www.ie.u-ryukyu.ac.jp/~wada/design07/contest2007.html
(2) http://www.ie.u-ryukyu.ac.jp/~wada/design07/conference.html

わだ・ともひさ
琉球大学工学部情報工学科教授
たなか・こういち
はやし・てるひこ
(株)ソリトンシステムズ

<筆者プロフィール>

和田知久．琉球大学工学部情報工学科教授，ニックネームは「ファイヤー和田」．8年前に大手電機メーカから沖縄にやってきて，楽しく活動中．最近は大容量ディジタル通信のOFDM方式受信器，複数アンテナを用いるアダプティブ信号処理からシステム実装，多数のプロセッサを用いて実装するソフトウェアラジオの広い範囲を専門としている．那覇にて，LSI設計ベンチャであるマグナデザインネットのチーフサイエンティストも務めており，南国にて生き生きとがんばっている．

田中孝一．DSPの黎明期にFAXモデムの開発に関わって以来，ディジタル信号処理に深くのめり込む．電子が隣の配線にトンネル効果で漏れてしまうほど電子回路の微細化が進み，電流，電圧といった古典的な概念が意味をなさなくなったとき，人類はキルヒホッフの法則の代わりに，どのような物理法則を用いてデバイスを動作させるのかといった近未来のことに興味を抱いている．

林輝彦．根っからの「ラジオ少年」は学校卒業後，何を血迷ったか，米国系のマイクロプロセッサ会社に就職してしまう．いくつかのCPUの開発に関わった後，今の会社に移り，シリコン・コンパイラ，HDLシミュレータ，アナログ合成などのEDAツールの普及に関わる．最近はアナログ設計屋に回帰しつつあり，ますます元気．